Numerical Computing - CSCI 3656--001
Homework 2 - 9/10/21
Kai Handelman

1.

$$f(x) = 4x^2 - 3x - 3$$

$$\Downarrow$$

$$x = \frac{-(-3) \pm \sqrt{(-3)^2 - 4(4)(-3)}}{2(4)}$$

$$= \frac{3 \pm \sqrt{9+48}}{8} = \frac{3 \pm \sqrt{57}}{8}$$

$$x = \frac{3+\sqrt{57}}{8}, \quad \frac{3-\sqrt{57}}{8}$$

2. *Code is attached in the submit - specifically the bisection function

```
(base) kai@rgnt1-38-189-dhcp HW2 % python3 hwScript.py
Start Interval: -2
End Interval: 1
Tolerance: 0.0001
Result from Bisection Method: -0.568756103515625
```

```
(base) kai@rgnt1-38-189-dhcp HW2 % python3 hwScript.py
Start Interval: 1
End Interval: 2
Tolerance: 0.0001
Result from Bisection Method: 1.31878662109375
```

3. *Left side is how I found g(x) and right side is proving that it fulfills the theorem.

$$f(x') = 4x^2 - 3x - 3 = 0$$

$$3x' + 3 = 4x'^2$$

$$x'\left(\frac{3x'+3}{4}\right) = (x'^2)x'$$

$$\frac{3x'^2 + 3x'}{4} = x'^3$$

$$x' = \sqrt[3]{\frac{3x'^2 + 3x'}{4}}$$

$$\therefore g(x) = \sqrt[3]{\frac{3x^2 + 3x}{4}}$$

f(x)'s roots:

$$a = \frac{3+\sqrt{57}}{8}, \quad b = \frac{3-\sqrt{57}}{8}$$

$$g(a) = a \quad g(b) = b$$

$$|g'(a)| \approx 0.52 < 1$$

$$|g'(b)| \approx -0.11 < 1$$

$\therefore$ g(x) coverges to r, (where $r = a, b$), at the rate of $|g'(r)|$, for initial guess near r.

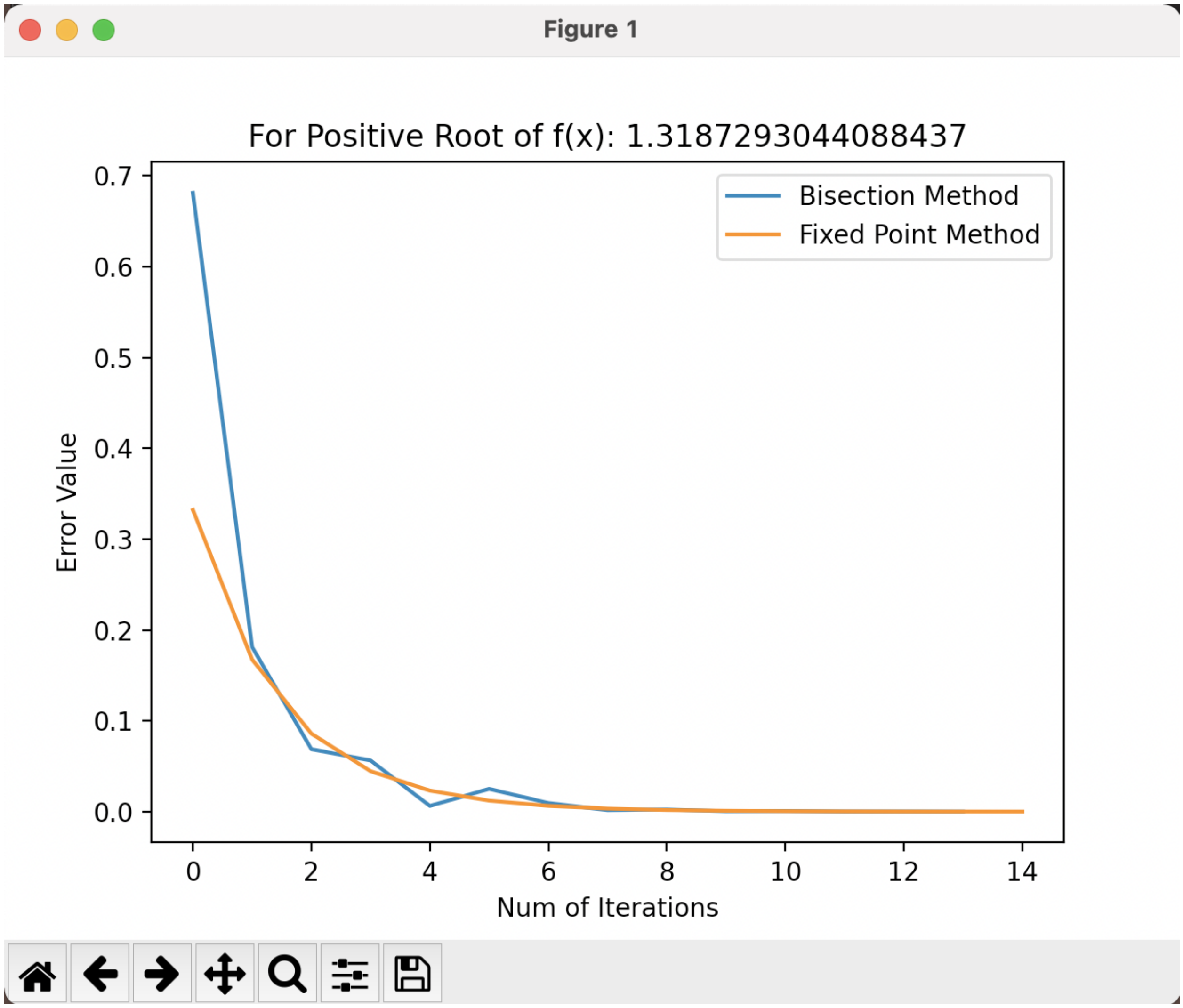
4. *Code is attached in the submit - specifically the fixedPoint function

```
Intital x_0 Guess: 0.1
Max Iterations Allowed: 5000
Tolerance of Error: 0.0001
Tolerance of Difference in Iteration: 1e-08
Iterations took: 28
Result: 1.3187292738222276
(base) kai@Kais-MacBook-Pro-2 HW2 %
```

Python can't find the negative root since it converts the number into a complex number during an exponent calculation of a negative root. Specifically, when I try to input f(x)'s negative root into a lambda function g(x), it returned: 0.2843646522044219+0.49253402549471176j

(*Will I continue to run into problems if I use python for this class?)

5.



Intervals used for Bisection: [1,3]
Tolerance used for Bisection: 10^{-4}

Initial Guess for x_0: 2
Tolerance of Error: 10^{-4}
Tolerance of Difference in Iteration: 10^{-8}
Maximum iterations: 100

I choose the initial intervals by picking one point from either side of the x-intercept to fulfill the root-finding theorem found in lecture 2.1. Due to the fact the bisection method iterates by getting a new interval via the equation:(b+a)/2. Since for my case, I had a = 1 and b=3, I decided to choose 2 as my initial guess for the Fixed Point method.

++++++++++

Code: Zip Link

```
import math
import matplotlib.pyplot as plt

def bisection(f, tol, a,b,target):
    error = []
    temp = None
    if f(a) * f(b) < 0:
        while abs((b-a) /2) > tol:
            temp = (a+b)/2
            if f(temp) == 0:
                return temp
            elif f(a) * f(temp) < 0:
                b = temp
            else:
                a = temp
            error.append(abs(target-temp))
        return ((a+b)/2,error)
    return (temp,error)

def fixedPoint(f, x, curTol, diffTol, M,target):
    error = []
    for b in range(M):
        temp = f(x)
        if curTol > abs(f(x)) or diffTol > abs(f(temp) - f(x)):
            return (x,b,error)
        x = temp
        error.append(abs(target-temp))
    return (x,M,error)


# For part 5
targetRoot = (3+math.sqrt(57))/8
func = lambda x: 4*(x**2) - 3*x -3
a = 1
b = 3
tol = 10**-4
(_,biErrors)=bisection(func,tol,a,b,targetRoot)
```

```
func = lambda x: ((3*(x**2)+3*x)/4)**(1./3)
x = 2
cT = 10**-4        #Tolerance of Error
cD = 10**-8        #Tolerance of Difference in Iteration
maxIter = 100
(_,_,fiError) = fixedPoint(func,x,cT,cD,maxIter,targetRoot)

biErrors = biErrors[0:15]
fiError = fiError[0:15]


line1,=plt.plot(biErrors,label="Bisection Method")
line2,=plt.plot(fiError,label="Fixed Point Method")
plt.title("For Positive Root of f(x): {}".format(targetRoot))
plt.legend(handles=[line1,line2])
plt.xlabel("Num of Iterations")
plt.ylabel("Error Value")
plt.show()



print(biErrors)
# # For bisection
# func = lambda x: 4*(x**2) - 3*x -3
# a = 1
# b = 2
# tol = 10**-4
# print("Start Interval: {}".format(a))
# print("End Interval: {}".format(b))
# print("Tolerance: {}".format(tol))
# print("Result from Bisection Method: {}".format(bisection(func,tol,a,b)))



# For fixedPoint
# t = 1/3
# func = lambda x: ((3*(x**2)+3*x)/4)**(1./3)
# temp = (3-math.sqrt(57))/8
```

```
# # print(temp)
# # print(func(temp))
# x = 0.1
# cT = 10**-4
# cD = 10**-8
# m = 5000
# print("Intital x_0 Guess: {}".format(x))
# print("Max Iterations Allowed: {}".format(m))
# print("Tolerance of Error: {}".format(cT))
# print("Tolerance of Difference in Iteration: {}".format(cD))
# r =fixedPoint(func,x,cT,cD,m,0)
# print("Iterations took: {}\nResult: {}".format(r[1],r[0]))
```